Proceedings Paper:
Zhou, Lei, Bai, Xiao, Zhang, Liang et al. (2 more authors) (2021) Fast Subspace Clustering Based on the Kronecker Product. In: Proceedings 25th International Conference on Pattern Recognition, ICPR 2021, Milan, Italy, January 10-15, 2021. International Conference on Pattern Recognition . , pp. 1558-1565.

https://doi.org/10.1109/ICPR48806.2021.9412287

Fast Subspace Clustering Based on the Kronecker Product

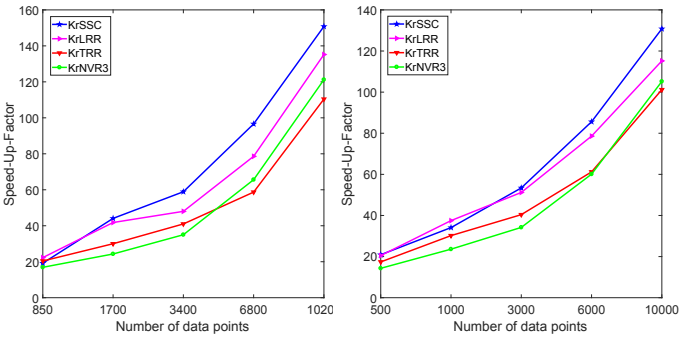
Lei Zhou*, Xiao Bai*, Liang Zhang*, Jun Zhou† and Edwin Hancock‡
*School of Computer Science and Engineering, Beihang University, Beijing, China
†School of Information and Communication Technology, Griffith University, Nathan, Australia
‡Department of Computer Science, University of York, York, U.K.

***Abstract*—Subspace clustering is a useful technique for many computer vision applications in which the intrinsic dimension of high-dimensional data is often smaller than the ambient dimension. Spectral clustering, as one of the main approaches to subspace clustering, often takes on a sparse representation or a low-rank representation to learn a block diagonal self-representation matrix for subspace generation. However, existing methods require solving a large scale convex optimization problem with a large set of data, with computational complexity reaches $\mathcal{O}(N^3)$ for N data points. Therefore, the efficiency and scalability of traditional spectral clustering methods can not be guaranteed for large scale datasets. In this paper, we propose a subspace clustering model based on the Kronecker product. Due to the property that the Kronecker product of a block diagonal matrix with any other matrix is still a block diagonal matrix, we can efficiently learn the representation matrix which is formed by the Kronecker product of k smaller matrices. By doing so, our model significantly reduces the computational complexity to $\mathcal{O}(kN^{3/k})$. Furthermore, our model is general in nature, and can be adapted to different regularization based subspace clustering methods. Experimental results on two public datasets show that our model significantly improves the efficiency compared with several state-of-the-art methods. Moreover, we have conducted experiments on synthetic data to verify the scalability of our model for large scale datasets.**

I. Introduction

In many computer vision applications, such as face recognition [1], [2], texture recognition [3] and motion segmentation [4], [5], visual data can be well characterized by subspaces. Moreover, the intrinsic dimension of high-dimensional data is often much smaller than the ambient dimension [6]. This has motivated the development of subspace clustering techniques which simultaneously cluster the data into multiple subspaces and also locate a low-dimensional subspace for each class of data.

Many subspace clustering algorithms have been developed during the past decade, including algebraic [7], [8], iterative [9], [10], statistical [11], [12], and spectral clustering methods [4], [2], [13], [14], [15], [16], [3], [17], [18], [19]. Among these approaches, spectral clustering methods have been intensively studied due to their simplicity, theoretical soundness, and empirical success. These methods are based on the self-expressiveness property of data lying in a union of subspaces. This states that each point in a subspace can be written as a linear combination of the remaining data points in that subspace. Two typical methods falling into this category are sparse subspace clustering (SSC) [4] and low-rank representation (LRR) [2]. SSC uses the ℓ_1 norm to encourage the sparsity of the self-representation coefficient matrix. LRR uses nuclear norm minimization to make the coefficient matrix low-rank.



(a) CMU PIE dataset (b) MNIST dataset

Fig. 1. Speed-up-factor of our Kronecker product based model over four baseline methods (SSC [4], LRR [2], TRR [20] and NVR3 [3]) (see Table I and Table II for details). It is evident that as size of dataset grows, the speed-up-factor significantly increases.

Motivated by SSC and LRR, some self-representation based methods have been developed, which use different regularization terms on the coefficient matrix. For example, least squares regression (LSR) [14] uses ℓ_2 regularization on the coefficient matrix. Correlation adaptive subspace segmentation (CASS) [13] uses a mixture of ℓ_1 and ℓ_2 regularization. Low-rank sparse subspace clustering (LRSSC) [21] and non-negative low-rank sparse (NNLRS) [22] construct regularization term as a blend of ℓ_1 and the nuclear norms. Because the nuclear norm does not achieve the accuracy in estimating the rank of real world data, subspace clustering with log-determinant approximation (SCLA) [17] replaces the nuclear norm used in LRR by non-convex rank approximations. Feature selection embedded subspace clustering (FSC) [18] reveals that not all features are equally important in the recovery of the low-dimensional subspaces. With feature selection both nuclear norm and non-convex rank approximations may give enhanced performance. Latent space sparse subspace clustering (LS3C) [15] seeks a linear projection of the data and learns a sparse representation in the projected latent low-dimensional space.

Despite the fact that SSC, LRR and their variants have

Manuscript 576 submitted to 2020 25th International Conference on Pattern Recognition. Received March 16, 2020.

achieved encouraging results in practice, they have critical limitations. In these approaches, the key idea is to learn a coefficient matrix which denotes the correlation between the data points. As the size of the coefficient matrix is N^2 for N data points, the SVD decomposition operation for solving the coefficient matrix has computational complexity of $\mathcal{O}(N^3)$. This is time consuming when the size of the data is large, so the efficiency of these approaches can not be guaranteed. Experiments in [19] and also in this paper show that some existing methods need to run for several hours on a normal computer when the number of test data reaches 10^4, which constrains the feasibility of these methods.

To overcome this limitation, we propose an efficient subspace clustering model based on the Kronecker product which achieves a significant reduction of computational complexity over quadratic [23]. Using the fact that each data point in a subspace can be written as a linear combination of all other points in that subspace, we can obtain points lying in the same subspace by learning the sparsest combination. Hence, in our model, we first learn a self-representation coefficient matrix formed by the Kronecker product of a series of small sparse matrices. Then we can constract a similarity matrix based on the coefficient matrix. Finally, a segmentation of the data can be obtained by spectral clustering on the similarity matrix.

The main contributions of this paper are as follows:

1) We propose an efficient subspace clustering model based on the Kronecker product. Our model uses the Kronecker product of a set of small matrices to build the self-representation coefficient matrix, which leads to a significant reduction of space and computational complexity.
2) Our model is adaptive for different regularization based subspace clustering methods [4], [2], [20], [3]. And we theoretically prove that the Kronecker product approximation in our model has good adaptivity.
3) Experimental results on large scale synthetic data and real world public datasets show that our method leads to a significant improvement in the clustering efficiency compared with the state-of-the-art methods while also achieving competitive accuracy.

II. Related Works

In this section, we review some classical and state-of-the-art methods for subspace clustering.

A. Sparse Subspace Clustering (SSC)

Given a data matrix $X = [x_i \in \mathbb{R}^D]_{i=1}^N$ that contains N data points drawn from n subspaces $\{S_i\}_{i=1}^n$. SSC [4] aims to find a sparse representation matrix C showing the mutual similarity of the points, i.e., $X = XC$. Since each point in S_i can be expressed in terms of the other points in S_i, such a sparse representation matrix C always exists. The SSC algorithm finds C by solving the following optimization problem:

$$\min_C \|C\|_1 \quad \text{s.t. } X = XC,\ diag(C) = 0, \tag{1}$$

where $diag(C) = 0$ eliminates the trivial solution.

B. Low-Rank Representation (LRR)

As pointed out in [2], SSC finds the sparsest representation of each data vector individually. There is no global constraint on its solution, so the SSC method may be inaccurate at capturing the global structures of data. Liu *et al.* [2] proposed that low rank can be a more appropriate criterion. Similar to SSC, LRR aims to find a low-rank representation of X by solving the following optimization problem, since the nuclear norm $\|C\|_*$ is the best convex approximation of $rank(W)$ over the unit ball of matrices:

$$\min_C \|C\|_* \quad \text{s.t. } X = XC, \tag{2}$$

where $\|C\|_*$ is the sum of the singular values of C.

C. Thresholding Ridge Regression (TRR)

The SSC and LRR methods solve the robust subspace clustering problem by removing the errors from the original data space and obtaining a good affinity matrix based on a clean dataset. Thus they need prior knowledge of the structure of the errors, which usually is unknown in practice. Peng *et al.* [20] proposed a robust subspace clustering method which overcomes this limitation by eliminating the effect of errors from the projection space with a model based on thresholding ridge regression (TRR):

$$\min_C \|X - XC\|_F^2 + \lambda\|C\|_F^2 \quad \text{s.t. } diag(C) = 0, \tag{3}$$

where λ is a balancing parameter and small values in C are truncated to zero by thresholding.

Based on TRR, a 2D nonlinear variance regularized ridge regression (NVR3) [3] was proposed to directly use 2D data, and thus the spatial information is maximally retained.

Each of these related works learns the coefficient matrix C with computational complexity $\mathcal{O}(N^3)$. This has limited the scalability of these methods on large scale datasets. Due to the effectiveness of the Kronecker product in reducing the computational complexity of matrix operations, we present a Kronecker product based subspace clustering model which can significantly improve the efficiency of the existing methods.

III. Kronecker Product Based Model

In this section, we describe our subspace clustering model based on the Kronecker product and develop an associated optimization scheme.

We first introduce the Kronecker product. Let $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{p\times q}$, the Kronecker product of matrices A and B is $A \otimes B \in \mathbb{R}^{mp\times nq}$ which is defined as:

$$A \otimes B = \begin{bmatrix} a_{11} \times B & \cdots & a_{1n} \times B \\ \vdots & \ddots & \vdots \\ a_{m1} \times B & \cdots & a_{mn} \times B \end{bmatrix},$$

where a_{ij} is the element at the i-th row and j-th column of A.

Manuscript 576 submitted to 2020 25th International Conference on Pattern Recognition. Received March 16, 2020.

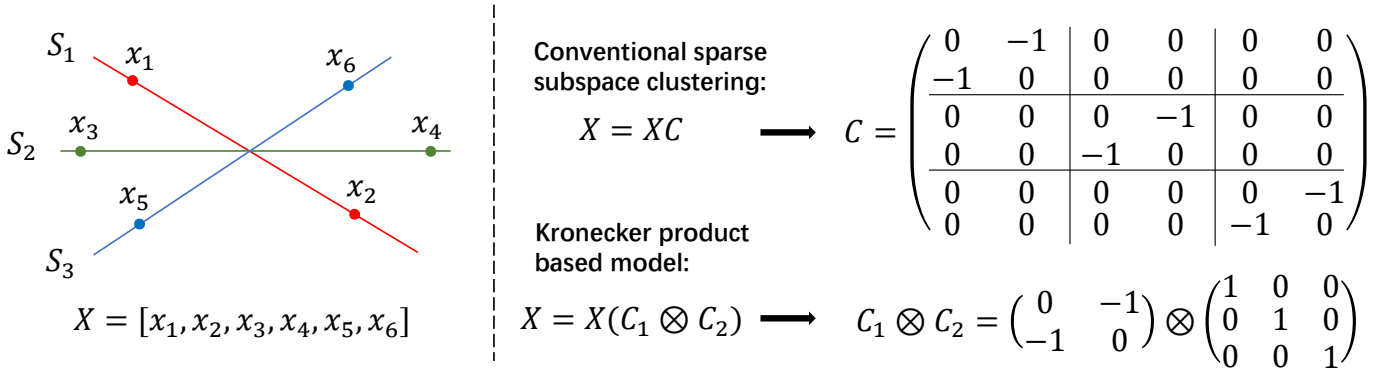


Fig. 2. Left: Three 1D subspaces in $\mathbb{R}^2$ with normalized data points. Right: The solutions of conventional sparse subspace clustering method (upper) and our Kronecker product based model (lower). As shown, the space and computational complexity of our model achieve significant reduction compared with conventional method.

A. Problem Statement and Formulation

Let $X = [x_i \in \mathbb{R}^D]_{i=1}^N \in \mathbb{R}^{D\times N}$ be a collection of data points drawn from different subspaces. The goal of subspace clustering is to find the segmentation of the points according to the subspaces. Based on the self-expressiveness property of data lying in a union of subspaces, i.e., each point in a subspace can be written as a linear combination of the remaining points in that subspace, we can obtain points lying in the same subspace by learning the sparsest combination. Therefore, we need to learn a sparse self-representation coefficient matrix C, where $X = XC$, and $C_{ij} = 0$ if the i-th and j-th data points are from different subspaces.

As our model aims to reduce the computational complexity with data size N, we rewrite X as $X = \{y_i^T \in \mathbb{R}^N\}_{i=1}^D$, where T denotes matrix transpose and $y_i \in \mathbb{R}^{N\times 1}$ is the i-th dimension of the data points. Without loss of generality, we assume that the self-representation matrix is formed by the Kronecker product of two smaller matrices C_1 and C_2, where $C_1 \in \mathbb{R}^{p_1\times q_1}$ and $C_2 \in \mathbb{R}^{p_2\times q_2}$, where $p_1p_2 = N$ and $q_1q_2 = N$. Here we use the important property that the Kronecker product of a block diagonal matrix with any other matrix is still a block diagonal matrix (as shown in Figure 2). We follow [20] to minimize the loss of self-representation. The optimization problem can be written as:

$$\min_{C_i} \|X - X(C_1 \otimes C_2)\|_F^2 + \lambda\|C_1 \otimes C_2\|_F^2, \tag{4}$$

where λ is a balancing parameter, and $\|.\|_F$ is the Frobenius norm.

B. Optimization

We solve problem (4) by updating each small matrix at a time, while keeping the other one fixed. Considering updating C_1, while C_2 fixed, we start by rewriting $\|X - X(C_1 \otimes C_2)\|_F^2$ as:

$$\begin{aligned}
&\|X - X(C_1 \otimes C_2)\|_F^2 \\
=&tr((X - X(C_1 \otimes C_2))^T(X - X(C_1 \otimes C_2))) \\
=&\|X\|_F^2 - 2tr(X(C_1 \otimes C_2)X^T) \\
&+ tr(X(C_1 \otimes C_2)(X(C_1 \otimes C_2))^T).
\end{aligned} \tag{5}$$

Since $\|X\|_F^2$ is a constant, let

$$\Phi = -2tr(X(C_1 \otimes C_2)X^T) + tr(X(C_1 \otimes C_2)(X(C_1 \otimes C_2))^T),$$

then, the problem that minimizing $\|X - X(C_1 \otimes C_2)\|_F^2$ is equivalent to minimizing Φ.

According to the block property of Kronecker product [24]:

$$a^T(C_1 \otimes C_2) = (vec(C_2^T M_{p_2,p_1}(a)C_1))^T,$$

where $a \in \mathbb{R}^N$ and $vec(X)$ forms a vector by column-wise stacking of the matrix X into a vector, and $M_{p_2,p_1}(a)$ reshapes a $p_1p_2 = N$ dimensional vector a to a $p_2 \times p_1$ matrix by extracting column from the vector a. Then

$$\begin{aligned}
\Phi =& \sum_{i=1}^D (-2y_i^T(C_1 \otimes C_2)y_i + y_i^T(C_1 \otimes C_2)(y_i^T(C_1 \otimes C_2))^T) \\
=& \sum_{i=1}^D (-2(vec(C_2^T M_{p_2,p_1}(y_i)C_1))^T y_i \\
&+ (vec(C_2^T M_{p_2,p_1}(y_i)C_1))^T vec(C_2^T M_{p_2,p_1}(y_i)C_1)).
\end{aligned} \tag{6}$$

Let $H_i = C_2^T M_{p_2,p_1}(y_i)$, $G_i = M_{q_2,q_1}(y_i)$. Then, using the property of trace that $tr(ABC) = tr(BCA)$ and $tr(A^T) = tr(A)$,

$$\begin{aligned}
\Phi =& \sum_{i=1}^D (-2tr((H_iC_1)^T G_i) + tr((H_iC_1)^T H_iC_1)) \\
=& \sum_{i=1}^D (-2tr(H_iC_1G_i^T) + tr((H_iC_1)^T H_iC_1)) \\
=& \sum_{i=1}^D (\|G_i - H_iC_1\|_F^2 - \|G_i\|_F^2).
\end{aligned} \tag{7}$$

Since $\|G_i\|_F^2$ is a constant, the optimization objective function of C_1 can be written as:

$$\min_{C_1} \|G - HC_1\|_F^2 + \lambda\|C_1\|_F^2 \tag{8}$$

where $H = \sum_{i=1}^D H_i$, $G = \sum_{i=1}^D G_i$. Eq. (8) is a well known ridge regression problem [25] whose optimal solution is $C_1 =$

Algorithm 1: Subspace Clustering Based on Kronecker Product.

Input: A set of data points $X = \{x_i\}_{i=1}^N$, the number of subspaces n, the number of small matrices k and the balance parameter λ.

Steps:

1. Learn the small matrices $C_1, C_2, \cdots, C_k$.

for $i = 1, ..., k$ **do**

Fix $C_1, \cdots, C_{i-1}, C_{i+1}, \cdots, C_k$, update C_i.

Optimize Eq. (8), estimate C_i by ridge regression solution.

end

2. Calculate the self-representation coefficient matrix C by the Kronecker product of small matrices, $C = \otimes_{i=1}^k C_i$.

3. Construct an affinity matrix by $W = |C| + |C|^T$.

4. Calculate the Laplacian matrix L of W.

5. Calculate the eigenvector matrix V of L corresponding to its n smallest nonzero eigenvalues.

6. Perform k-means clustering algorithm on the rows of V.

Output: The clustering result of X.

$(H^T H + \lambda I)^{-1} H^T G$. We can solve C_2 in a similar manner to C_1, when C_1 is fixed. As $H \in \mathbb{R}^{q_2 \times p_1}$, $G \in \mathbb{R}^{q_2 \times q_1}$ and $p_1 p_2 = N, q_1 q_2 = N$, the computational complexity for this solution is $\mathcal{O}(2N^{3/2})$.

When the number of small matrices is k, we can also solve it by updating one small matrix at a time, while keeping the remaining matrices fixed. In this situation, the problem is the same as $k = 2$ solved above. As $\prod_{i=1}^k p_i = N$, $\prod_{i=1}^k q_i = N$, then the computational complexity of the whole optimization is $\mathcal{O}(kN^{3/k})$.

We have obtained the optimal solution of self-representation coefficient matrix $C = \otimes_{i=1}^k C_i$, where $C_{ij} = 0$ if the i-th and j-th data points are from different subspaces. Hence, the affinity matrix W can be defined as $W = |C| + |C|^T$, where $|C|$ denotes the absolute value matrix of C. Then the segmentation of the data X in different subspaces can be obtained by applying a spectral clustering algorithm to the affinity matrix W. The whole Kronecker product based subspace clustering model is summarized in Algorithm 1.

IV. Theoretical Analysis

In this section, we give a theoretical analysis of our Kronecker product based model, including a) the adaptivity on different regularizations, b) theoretical convergence analysis, c) complexity analysis.

A. Adaptivity on Different Regularizations

Since many self-representation based methods use different regularizations on the coefficient matrix, we show that our model can be applied to a variety of different regularizations. We refer to our subspace clustering method described in Section III as KrTRR (Kronecker product based TRR). It utilizes the Frobenius norm to regularize the coefficient matrix. In Eq. (8), we simplify the sparsity constraint from $\|C_1 \otimes C_2\|_F^2$ to $\|C_1\|_F^2$, using the Kronecker product lemma:

Lemma 1. *Let* $C = C_1 \otimes C_2$, *then* $\|C\|_F^2 = \|C_1\|_F^2 \|C_2\|_F^2$.

Proof. Assume C^{ij} is the i-th column j-th row element in C, $C_1 \in \mathbb{R}^{m \times n}$, $C_2 \in \mathbb{R}^{p \times q}$, $C \in \mathbb{R}^{mp \times nq}$. Then $\|C\|_F^2 = \|C_1 \otimes C_2\|_F^2 = \sum_{i=1}^m \sum_{j=1}^n \|C_1^{ij} C_2\|_F^2 = \sum_{i=1}^m \sum_{j=1}^n (C_1^{ij})^2 \|C_2\|_F^2 = \|C_1\|_F^2 \|C_2\|_F^2$. □

Here we introduce two additional Kronecker product lemmas to show that our model can be applied to alternative regularizations.

Lemma 2. *Let* $C = C_1 \otimes C_2$, *then* $\|C\|_1 = \|C_1\|_1 \|C_2\|_1$.

Proof. Assume C^{ij} is the i-th column j-th row element in C, $C_1 \in \mathbb{R}^{m \times n}$, $C_2 \in \mathbb{R}^{p \times q}$, $C \in \mathbb{R}^{mp \times nq}$. Then $\|C\|_1 = \|C_1 \otimes C_2\|_1 = \sum_{i=1}^m \sum_{j=1}^n \| |C_1^{ij}| C_2 \|_1 = \sum_{i=1}^m \sum_{j=1}^n |C_1^{ij}| \|C_2\|_1 = \|C_1\|_1 \|C_2\|_1$. □

Lemma 3. *Let* $C = C_1 \otimes C_2$, *then* $\|C\|_* = \|C_1\|_* \|C_2\|_*$.

Proof. Assume the SVD decompositions of C_1 and C_2 are $C_1 = U_1 \Sigma_1 V_1^T$ and $C_2 = U_2 \Sigma_2 V_2^T$, respectively. Then $\|C_1\|_*$ is the sum of nonzero entries in the diagonal matrix Σ_1, $\|C_2\|_*$ is the sum of nonzero entries in the diagonal matrix Σ_2. $C = C_1 \otimes C_2 = (U_1 \Sigma_1 V_1^T) \otimes (U_2 \Sigma_2 V_2^T) = (U_1 \otimes U_2)((\Sigma_1 V_1^T) \otimes (\Sigma_2 V_2^T)) = (U_1 \otimes U_2)(\Sigma_1 \otimes \Sigma_2)(V_1 \otimes V_2)^T$. Because $\Sigma_1 \otimes \Sigma_2$ is a diagonal matrix, then the SVD decomposition of C is $C = (U_1 \otimes U_2)(\Sigma_1 \otimes \Sigma_2)(V_1 \otimes V_2)^T$. So that $\|C\|_*$ is the sum of nonzero entries in the diagonal matrix $\Sigma_1 \otimes \Sigma_2$ which is the product of the sum of nonzero entries in the diagonal matrix Σ_1 and Σ_2. Then $\|C\|_* = \|C_1\|_* \|C_2\|_*$. □

Based on these two lemmas, the ℓ_1 norm and nuclear norm regularizations on the coefficient matrix $\| \otimes_{i=1}^k C_i\|_1$, $\| \otimes_{i=1}^k C_i\|_*$ can be simplified to $\|C_i\|_1$ and $\|C_i\|_*$ as shown in Eq. (8). So we can also utilize the ℓ_1 norm and nuclear norm on the self-representation coefficient matrix with a manner similar to SSC and LRR, i.e.

$$\min_{C_i} \|X - X(\otimes_{i=1}^k C_i)\|_F^2 + \lambda \| \otimes_{i=1}^k C_i\|_1 \tag{9}$$

and

$$\min_{C_i} \|X - X(\otimes_{i=1}^k C_i)\|_F^2 + \lambda \| \otimes_{i=1}^k C_i\|_* \tag{10}$$

We refer to these two methods as KrSSC and KrLRR. Following [3], we can preprocess the data by 2DPCA [26] to retain the spatial information in the 2D data. Then we can use the KrTRR method to learn the coefficient matrix C as done in [3]. We refer to this method as KrNVR3. The optimization of these variants of the Kronecker product based method are essentially the same as KrTRR.

In summary, we can leverage the Kronecker product to reduce the computational complexity of learning the coefficient matrix with different regularization options, e.g. Frobenius norm, ℓ_1 norm and nuclear norm. We present four methods KrSSC, KrLRR, KrTRR and KrNVR3 based on different

Manuscript 576 submitted to 2020 25th International Conference on Pattern Recognition. Received March 16, 2020.

regularizations and compare them with baseline methods in Section V.

B. Theoretical Convergence Analysis

Here, we prove the reliability of Kronecker product approximation using a theoretical convergence analysis.

According to the idea of mathematical induction, we consider the special condition that $k = 2$ to approximate a $p^2 \times p^2$ matrix C by $A \otimes A$, where A is a $p \times p$ matrix. The matrix C is partitioned into p^2 matrices with dimension $p \times p$, i.e.

$$C = \begin{bmatrix} C_{11} & \cdots & C_{1p} \\ \vdots & \ddots & \vdots \\ C_{p1} & \cdots & C_{pp} \end{bmatrix}$$

Let

$$C^* = [vec(C_{11}), vec(C_{12}), \cdots, vec(C_{pp})]$$

Then, we can denote the approximate loss function by:

$$\begin{aligned} l &= tr(A \otimes A - C)^2 \\ &= (tr(A)^2)^2 - 2a^T C^* a + tr(C)^2 \end{aligned} \tag{11}$$

where $a = vec(A)$. Since

$$\begin{aligned} tr(A \otimes A)C &= (vec(C))^T vec(A \otimes A) \\ &= (vec(C^*))^T (vec(A) \otimes vec(A)) \\ &= (vec(C^*))^T vec((vec(A)(vec(A))^T) \\ &= trC^*(vec(A))(vec(A))^T \\ &= (vec(A))^T C^* vec(A) \\ &= a^T C^* a \end{aligned} \tag{12}$$

Let $\nu(A)$ be the vector with non-duplicate elements of $vec(A)$ and $a = vec(A) = D\nu(A)$, here D is the duplication matrix. Then, the first differential of l is

$$\begin{aligned} \mathrm{d}l &= 4(tr(A)^2)a^T \mathrm{d}a - 4a^T C^* \mathrm{d}a \\ &= 4(tr(A)^2)a^T D\mathrm{d}\nu(A) - 4a^T C^* D\mathrm{d}\nu(A) \end{aligned} \tag{13}$$

The first derivative is

$$\frac{\partial l}{\partial \nu(A)} = 4(tr(A)^2)D^T vec(A) - 4D^T C^* vec(A) \tag{14}$$

Then, we obtain the first-order condition

$$tr(A)^2 vec(A) = C^* vec(A) \tag{15}$$

This is an eigenvalue problem in terms of C^*. The vector a minimizing Eq. (11) must be proportional to the eigenvector corresponding to the largest eigenvalue of C^*. In other words, for an arbitrary matrix with any dimension, we can partition it based on the dimensions of small matrices needed to approximate the large matrix via Kronecker product. moreover, the small matrices always have a convergent solution through the largest eigenvector of the partitioned large matrix. This means that the technique used to approximate the large self-representation matrix by the Kronecker product of small matrices in our model is reliable.

C. Complexity Analysis

Here we discuss the space memory requirement and computational complexity of our Kronecker product based methods and compare it to the relevant methods in the literature. When the data size is N, methods in [4], [2], [20], [3] need to solve the self-representation coefficient matrix C with the dimension $N \times N$, i.e., the memory space complexity of these methods is $\mathcal{O}(N^2)$. But in our work, we leverage the Kronecker product of a set of small matrices to approximate the self-representation coefficient matrix C. When the number of small matrices is k, the size of small matrices is $N^{2/k}$. Thus, the space complexity of our methods is $\mathcal{O}(kN^{2/k})$.

For learning process the self-representation coefficient matrix C with size N^2, existing methods use a SVD decomposition operation whose computational complexity is $\mathcal{O}(N^3)$. As our methods update one small matrix at a time, and the size of the small matrix is $N^{2/k}$, we achieve $\mathcal{O}(kN^{3/k})$ computational complexity. Since $N^{1/k} \ll N$ when $k > 1$, there is significant reduction in both the memory space and computational complexity compared with the existing methods. This efficiency gain is achieved by using the Kronecker product.

V. Experiments

We have conducted three sets of experiments on both real and synthetic datasets to verify the effectiveness of the proposed methods. Several state-of-the-art or classical spectral subspace clustering methods were taken as the baseline algorithms. These included sparse subspace clustering (SSC) [4], low-rank representation (LRR) [2], thresholding ridge regression (TRR) [20], and nonlinear variance regularized ridge regression (NVR3) [3]. In the experiments, we used the codes provided by the respective authors for computing the self-representation matrix C, where the parameters were tuned to give the best clustering accuracy. Then we applied the normalized spectral clustering in [27] to the affinity matrix $W = |C| + |C|^T$.

Evaluation criteria: we used both the clustering accuracy and running time of the whole clustering process to evaluate the performance of the subspace clustering methods, where the clustering accuracy is calculated as

$$\text{clustering accuracy} = \frac{\text{\# of correctly classified points}}{\text{total \# of points}} \times 100$$

In all our experiments, the clustering accuracy and running time were averaged over 10 trials. All experiments were implemented with MATLAB code and ran on a PC with Intel Core-i7 3.6GHz CPU, 32GB RAM.

A. Face Clustering

As subspaces are commonly used to capture the appearance of faces under varying illuminations, we test the performance of our method on face clustering with the CMU PIE database [28]. The CMU PIE database contains 41,368 images of 68 people under 13 different poses, 43 different illumination conditions, and 4 different expressions. In our experiment,

Manuscript 576 submitted to 2020 25th International Conference on Pattern Recognition. Received March 16, 2020.

TABLE I
THE AVERAGE RUNNING TIME (SECONDS) AND CLUSTERING ACCURACY ON THE CMU PIE DATABASE WITH DIFFERENT NUMBER OF OBJECTS. EACH OBJECT CONSISTS OF 170 FACE IMAGES UNDER DIFFERENT ILLUMINATIONS AND EXPRESSIONS. '-' DENOTES THAT THE COMPUTATIONAL COST IS UNACCEPTABLE FOR OUR PC, DUE TO THE MEMORY AND TIME LIMIT.

No. Objects	5 Objects		10 Objects		20 Objects		40 Objects		60 Objects	
	Time	Acc.	Time	Acc.	Time	Acc.	Time	Acc.	Time	Acc.
SSC	243.6	92.47	1182	89.25	3618	84.31	14502	82.37	-	-
KrSSC	**12.7**	91.28	**26.8**	88.27	**61.4**	83.86	**150.2**	81.75	**274.3**	79.48
LRR	216.4	94.53	852.5	92.14	2743	89.21	11463	85.47	-	-
KrLRR	**9.7**	92.51	**20.4**	90.72	**57.2**	88.13	**145.8**	85.21	**254.8**	83.65
TRR	152.7	97.35	548.2	96.05	2167	94.54	8427	91.74	-	-
KrTRR	**7.5**	95.21	**18.3**	94.52	**52.8**	93.84	**143.5**	90.23	**260.1**	87.26
NVR3	190.5	98.51	624.6	97.51	2536	95.75	11826	93.15	-	-
KrNVR3	**11.3**	97.14	**25.7**	96.26	**72.4**	93.96	**180.4**	91.57	**312.5**	89.15

we used the face images in five near frontal poses (P05, P07, P09, P27, P29). Then each people has 170 face images under different illuminations and expressions. Each image was manually cropped and normalized to a size of 32×32 pixels. In each experiment, we randomly picked $n \in \{5, 10, 20, 40, 60\}$ individuals to investigate the performance of the proposed method. For our models, we set the number of small matrices $k = 2$ and $\lambda = 0.25$. For different number of objects n, we randomly chose n people with 10 trials and took all the images of them as the subsets to be clustered. Then we conducted experiments on all 10 subsets and report the average running time and clustering accuracy with a different number of objects in Table I.

In the original work, SSC, LRR, TRR, and NVR3 all test on a small subset which consists of no more than 1,000 data points. Because of the memory and time limit, these methods can not run on a dataset of size $\mathcal{O}(10^4)$. In our experiment, the data size is in the range of $N \in \{850, 1700, 3400, 6800, 10200\}$, corresponding to 5-60 objects per face. As shown in Table I, the efficiency of all alternative methods degrades drastically when N increases. When $N > 10000$ (60 objects), the space and computational complexity of these methods are unacceptable for our PC. In contrast, the computational time of Kronecker product based methods is significantly lower compared with the corresponding approaches. Our methods can easily handle more than 10,000 data points with an acceptable computing time. Further, we can see from Table I that the Kronecker product based methods also obtain competitive clustering accuracy (down 2 percent at most). This suggests that our model is potentially more suitable than previous methods on large scale dataset for real world applications.

B. Handwritten Digit Clustering

Database of handwritten digits is also widely used in subspace learning and clustering. We test the proposed methods on handwritten digit clustering with the MNIST dataset [29]. This dataset contains 10 clusters, including handwritten digits 0-9. Each cluster contains 6,000 images for training and 1,000 images for testing, with a size of 28×28 pixels in each image. We used all the 70,000 handwritten digit images for subspace clustering. Different from the experimental settings for face clustering, we fixed the number of clusters $n = 10$ and chose different number of data points for each cluster with 10 trials. Each cluster contains N_i data points randomly chosen from corresponding 7,000 images, where $N_i \in \{50, 100, 1000, 3000, 7000\}$, so that the number of points $N \in \{500, 1000, 10000, 30000, 70000\}$. Then we applied all methods on this dataset for comparison. For our models, we set the number of small matrices $k = 2$ and $\lambda = 0.2$. The average running time and clustering accuracy with different number of data points are shown in Table II.

It can be seen that the efficiency of KrSSC, KrLRR, KrTRR and KrNVR3 significantly outperform the corresponding baseline methods, which indicates the effectiveness of the Kronecker product method proposed in this paper. Table II also shows that our method and its variants obtain competitive clustering accuracy compared with the corresponding baseline methods.

TABLE IV
THE AVERAGE RUNNING TIME AND CLUSTERING ACCURACY OF OUR METHODS WITH DIFFERENT k.

k	2	3	4	5
average running time (seconds):				
KrSSC	715.6	285.7	61.2	25.4
KrLRR	682.5	274.3	52.7	20.6
KrTRR	755.1	314.2	84.3	31.5
KrNVR3	794.3	321.5	91.6	36.2
average clustering accuracy:				
KrSSC	83.14	81.85	75.42	67.25
KrLRR	84.43	82.20	77.16	68.17
KrTRR	90.75	89.06	84.27	73.41
KrNVR3	92.54	90.62	85.34	75.24

C. Large-Scale Experiment

To verify the scalability of our method on large scale datasets, we also ran experiments on synthetic da-

TABLE II
THE AVERAGE RUNNING TIME (SECONDS) AND CLUSTERING ACCURACY ON THE MNIST DATASET WITH DIFFERENT NUMBER OF DATA POINTS. THE DATA CONSISTS OF RANDOMLY CHOSEN $N_i \in \{50, 100, 1000, 3000, 7000\}$ IMAGES FOR EACH OF THE 10 DIGITS. '-' DENOTES THAT THE COMPUTATIONAL COST IS UNACCEPTABLE ON OUR PC DUE TO THE MEMORY AND TIME COST.

No. Points	500		1000		10000		30000		70000	
	Time	Acc.	Time	Acc.	Time	Acc.	Time	Acc.	Time	Acc.
SSC	152.4	83.36	638.2	82.45	-	-	-	-	-	-
KrSSC	**7.3**	81.25	**18.7**	81.17	**192.4**	79.42	**411.5**	76.15	**683.2**	73.34
LRR	145.5	85.75	614.8	85.14	-	-	-	-	-	-
KrLRR	**7.1**	83.24	**16.4**	83.20	**160.8**	81.52	**384.5**	79.21	**641.5**	76.53
TRR	113.2	90.28	476.4	89.78	-	-	-	-	-	-
KrTRR	**6.5**	88.95	**15.8**	88.65	**168.2**	85.76	**403.8**	83.26	**795.6**	81.53
NVR3	118.5	91.85	531.1	91.28	-	-	-	-	-	-
KrNVR3	**8.3**	90.08	**22.5**	90.14	**243.6**	86.27	**627.5**	83.87	**968.4**	82.41

TABLE III
THE AVERAGE RUNNING TIME (SECONDS) AND CLUSTERING ACCURACY ON SYNTHETIC DATASET WITH DIFFERENT NUMBER OF DATA POINTS. THE DATA CONSISTS OF RANDOMLY CHOSEN $N_i \in \{100, 1000, 2000, 10000, 20000\}$ POINTS FOR EACH OF THE 5 SUBSPACES. '-' DENOTES THAT THE COMPUTATIONAL COST IS UNACCEPTABLE FOR OUR PC DUE TO THE MEMORY AND TIME LIMIT.

No. Points	500		5000		10000		50000		100000	
	Time	Acc.	Time	Acc.	Time	Acc.	Time	Acc.	Time	Acc.
SSC	135.4	94.15	1824	93.86	5413	91.05	-	-	-	-
KrSSC	**6.2**	92.12	**53.4**	91.18	**164.2**	89.73	**231.5**	85.04	**285.7**	81.85
LRR	118.6	95.27	1645	94.57	4853	92.14	-	-	-	-
KrLRR	**6.0**	93.24	**49.3**	92.21	**152.7**	89.49	**216.2**	86.03	**274.3**	82.20
TRR	89.5	98.85	1627	97.15	5825	95.69	-	-	-	-
KrTRR	**5.9**	98.06	**46.7**	96.53	**185.3**	95.05	**250.3**	93.16	**314.2**	89.06
NVR3	96.4	99.91	1752	98.61	6024	97.10	-	-	-	-
KrNVR3	**6.0**	99.07	**52.8**	98.11	**207.5**	96.24	**260.1**	93.89	**321.5**	90.62

ta. Following [19], we randomly generated $n = 5$ subspaces, each of dimension $d = 6$ in an ambient space of dimension $D = 9$. Each subspace contains N_i data points randomly generated on the unit sphere, where $N_i \in \{100, 1000, 2000, 10000, 20000\}$, so that the number of points $N \in \{500, 5000, 10000, 50000, 100000\}$. Due to the memory and time limit, SSC, LRR, TRR and NVR3 were run for $N \leq 10000$. For our models, $\lambda = 0.2$, the number of small matrices $k = 2$ for $N \in \{500, 5000, 10000\}$ and $k = 3$ for $N \in \{50000, 100000\}$. With different number of sample points, we conducted experiments on all methods and report the average running time and clustering accuracy in Table III.

As shown in Table III, the advantage of our method and its variants over the baseline methods is more marked on large scale datasets. When the dataset size reaches 10,000, the computational running time of the alternate methods under comparison are about two hours each, but our Kronecker product based methods only need a few thousand seconds even for 100,000 data points. From Table III, it is also clear that when k increases from 2 to 3 for $N \in \{50000, 100000\}$, the running time decreases significantly. The clustering accuracy can also be guaranteed compared with existing methods. Due to the limitations of memory space and computational complexity, the alternative methods can not be applied to a dataset of larger than 10,000 points. This again suggests that our methods are potentially more suitable for large real world applications.

D. Parameter Sensitivity

Here, we report experimental results on a synthetic dataset to illustrate the sensitivity of the Kronecker product based methods to parameter variations. As the parameters k (number of the small matrices) and λ (the balancing parameter of Eq. (4)) in our model are both related to the dataset size N, we fix $N = 100000$. Table IV shows the average running time and clustering accuracy of our methods with different $k \in \{2, 3, 4, 5\}$. We can see that when k increases, the running time significantly decreases but with the sacrifice of clustering accuracy. This implies that the number of small matrices k should be determined by the size of dataset with a compromise between efficiency and accuracy. Figure 3 shows the clustering accuracy of our methods with different balance parameter λ. It is evident that the clustering accuracy is insensitive when $\lambda \in (0.1, 0.5)$.

VI. CONCLUSION

We have presented a fast subspace clustering model based on the Kronecker product. Due to the property that the Kronecker product of a block diagonal matrix and any other matrix

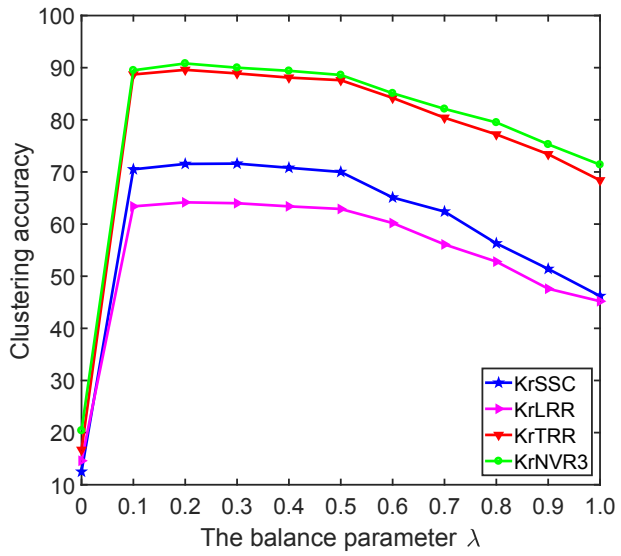


Fig. 3. The average clustering accuracy of our methods with different balance parameter λ.

is still a block diagonal matrix, we learn the representation matrix of spectral clustering using the Kronecker product of a set of smaller matrices. Thanks to the superiority of the Kronecker product in reducing the computational complexity of matrix operations, the memory space and computational complexity of our methods achieve significant efficiency gain compared with several baseline approaches (SSC, LRR, TRR, and NVR3). We have presented four variants of the Kronecker product based method, namely KrSSC, KrLRR, KrTRR and KrNVR3. Experimental results on face clustering and handwriting digit clustering show that our methods achieve significantly improvement in efficiency compared with the state-of-the-art methods. Moreover, we have presented results on synthetic data which has verified the scalability of our methods on large scale datasets.

REFERENCES

[1] R. Basri and D. W. Jacobs, "Lambertian reflectance and linear subspaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 2, pp. 218–233, 2003.

[2] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma, "Robust recovery of subspace structures by low-rank representation." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 171–184, 2013.

[3] C. Peng, Z. Kang, and Q. Cheng, "Subspace clustering via variance regularized ridge regression," in *Computer Vision and Pattern Recognition*, 2017.

[4] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2765–2781, 2013.

[5] K.-i. Kanatani, "Motion segmentation by subspace separation and model selection," in *International Conference on Computer Vision*, vol. 2, 2001, pp. 586–591.

[6] R. Vidal, "Subspace clustering," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 52–68, 2011.

[7] J. P. Costeira and T. Kanade, "A multibody factorization method for independently moving objects," *International Journal of Computer Vision*, vol. 29, no. 3, pp. 159–179, 1998.

[8] R. Vidal, Y. Ma, and S. Sastry, "Generalized principal component analysis (gpca)," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 12, pp. 1945–1959, 2005.

[9] P. K. Agarwal and N. H. Mustafa, "K-means projective clustering," in *Symposium on Principles of Database Systems*, 2004, pp. 155–165.

[10] P. S. Bradley and O. L. Mangasarian, "K-plane clustering," *Journal of Global Optimization*, vol. 16, no. 1, pp. 23–32, 2000.

[11] S. R. Rao, R. Tron, R. Vidal, and Y. Ma, "Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories," in *Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[12] M. E. Tipping and C. M. Bishop, "Mixtures of probabilistic principal component analyzers," *Neural computation*, vol. 11, no. 2, pp. 443–482, 1999.

[13] C. Lu, J. Feng, Z. Lin, and S. Yan, "Correlation adaptive subspace segmentation by trace lasso," in *International Conference on Computer Vision*, 2014, pp. 1345–1352.

[14] C. Y. Lu, H. Min, Z. Q. Zhao, L. Zhu, D. S. Huang, and S. Yan, "Robust and efficient subspace segmentation via least squares regression," in *European Conference on Computer Vision*, 2012, pp. 347–360.

[15] V. M. Patel, H. Van Nguyen, and R. Vidal, "Latent space sparse subspace clustering," in *International Conference on Computer Vision*, 2013, pp. 225–232.

[16] V. M. Patel and R. Vidal, "Kernel sparse subspace clustering," in *International Conference on Image Processing*, 2014, pp. 2849–2853.

[17] C. Peng, Z. Kang, H. Li, and Q. Cheng, "Subspace clustering using log-determinant rank approximation," in *International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 925–934.

[18] C. Peng, Z. Kang, M. Yang, and Q. Cheng, "Feature selection embedded subspace clustering," *IEEE Signal Processing Letters*, vol. 23, no. 7, pp. 1018–1022, 2016.

[19] C. You, D. Robinson, and R. Vidal, "Scalable sparse subspace clustering by orthogonal matching pursuit," in *Computer Vision and Pattern Recognition*, 2016, pp. 3918–3927.

[20] X. Peng, Z. Yi, and H. Tang, "Robust subspace clustering via thresholding ridge regression," in *AAAI Conference on Artificial Intelligence*, 2015, pp. 3827–3833.

[21] L. Zhuang, H. Gao, Z. Lin, Y. Ma, X. Zhang, and N. Yu, "Non-negative low rank and sparse graph for semi-supervised learning," in *Computer Vision and Pattern Recognition*, 2012, pp. 2328–2335.

[22] Y. X. Wang, H. Xu, and C. Leng, "Provable subspace clustering: when lrr meets ssc," in *Advances in Neural Information Processing Systems*, 2013, pp. 64–72.

[23] C. F. Van Loan and N. Pitsianis, "Approximation with kronecker products," in *Linear algebra for large scale and real-time applications*. Springer, 1993, pp. 293–314.

[24] C. F. Van Loan, "The ubiquitous kronecker product," *Journal of computational and applied mathematics*, vol. 123, no. 1, pp. 85–100, 2000.

[25] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[26] J. Yang, D. Zhang, A. F. Frangi, and J.-y. Yang, "Two-dimensional pca: a new approach to appearance-based face representation and recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 1, pp. 131–137, 2004.

[27] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[28] T. Sim, S. Baker, and M. Bsat, "The cmu pose, illumination, and expression (pie) database of human faces," Pittsburgh, PA, Tech. Rep. CMU-RI-TR-01-02, January 2001.

[29] Y. Lcun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

Manuscript 576 submitted to 2020 25th International Conference on Pattern Recognition. Received March 16, 2020.